Mauve: a Component-based Modeling Framework for Real-time Analysis of Robotic Applications.

Charles Lesire, David Doose, Hugues Cassé

Mauve: a Component-based Modeling Framework for Real-time Analysis of Robotic Applications

Charles Lesire and David Doose and Hugues Cassé

I. Introduction

Robots are more and more used in very diverse situations (services to persons, military missions, crisis management, ...) in which robots must give some guarantees of *safety* and *reliability*. To be really integrated in everyday life, robots must fulfil some requirements. Among these requirements, we focus on the non-functional requirements on embedded software [1], and more specifically on real-time software requirements. These requirements are most of the time fulfilled by proving the *schedulability* of the embedded software.

Analysing and validating such properties on an existing hand-coded software requires some reverse modelling of the software, leading to approximations of its behaviour. These approximations may have certification authorities not be confident on the robot dependability.

This paper proposes an integrated development methodology that starts from software component modelling, and leads to both validation of the embedded software and generation of deployable embedded software.

II. Motivations

Component-Based Software Engineering (CBSE) is an essential design paradigm for robotic software development [2], applied in many applications [3], [4], [5], [6]. Resulting components are reusable and composable, which ease their development, the architecture deployment, and its validation.

Recent practices in robotic software development intensively use middlewares (such as Orocos [7], Genom [8] or ROS [9]) to help engineers to focus on the robotic aspects of development without having to manipulate low-level OS primitives for task management or communication policies.

This paper presents a methodology for robotic application design that first requires a specification of the robotic software architecture inspired by both CBSE and separation of concerns [10]. Then, runtime execution is ensured by the generation of embedded code for the Orocos middleware, which is renowned for its real-time skills.

Although such model-driven engineering frameworks targeting Orocos already exist (BRIDE [11] – developed in the BRICS project [12] –, Proteus [13]), the methodology proposed in this paper focuses on real-time validation of the robotic application, by directly analysing the architecture specification, and limiting the middleware usage to its verifiable subset.

III. Mauve

The Mauve (Modeling Autonomous Vehicles) methodology settles on a formal model, that clearly separates the component Computation (modelled as *codels* [8]), and the component and system Configuration, modelled through a *validable* formal language. Finally, the schedulable software architecture is deployed by targeting a subset of the Orocos middleware [7] that offers OS abstraction and portability of the application.

The Mauve implementation relies on a domain specific language (DSL, partially represented on Fig. 2(a)). The abstract syntax of the Mauve DSL is used by the real-time analysis. The Mauve framework also provide a concrete syntax of the Mauve DSL used by engineers to specify their architecture (Fig. 2(b)).

The complete process is shown on Fig. 1 and is detailed in this section.

A. Codels

Codels, that stand for "elementary codes", concern the *computation* of a component, i.e. data processing and algorithms. A codel is a function directly implemented in a programming language, without any dependence neither on the component specification, nor on the targeted middleware, OS or hardware architecture. This separation makes codels framework-independent, reusable, maintenable, and shareable [14].

B. Component model

Mauve allows to model libraries of reusable components. A component is described by several elements:

- Properties: each component has a set of properties that define the component parameters;
- Methods: each component can provide services to the other components or require specific services from other, unknown components;
- Ports: components can exchange data through ports; a component defines its input and output ports, with the type of the received or sent data;

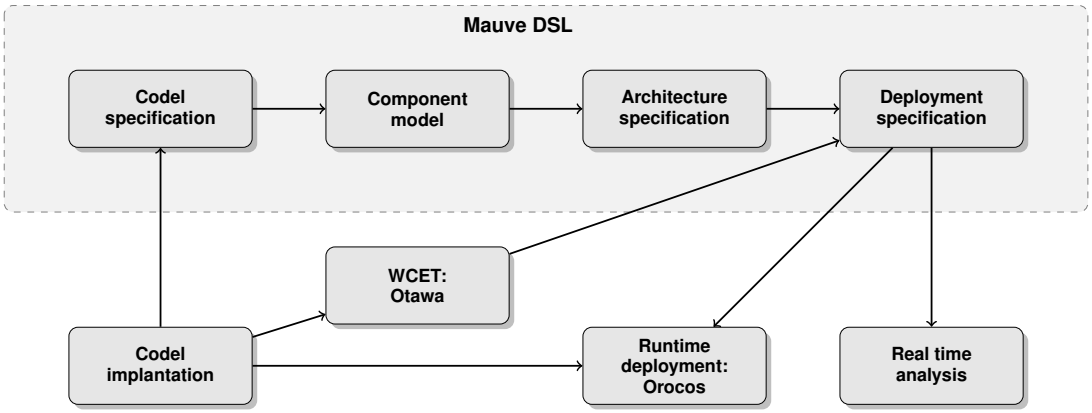


Fig. 1: The Mauve process

- Behaviour: each component has a specific behaviour defined by a Finite State Machine (FSM); A state machine is made of states and transitions between states.

Mauve also contains a simple expression language with assignments, conditional instructions, method calls, codel execution, etc. This expression language is used to specify the behaviour of a component:

- the entry expression is executed at the step when a state is entered;
- the run expression is executed at each step when a state is active;
- the exit expression is executed at the step when a state is left.

This component description is consistent with a lot of component-based middlewares classically used in robotics [3], [8], [7], [9].

C. Architecture specification

The specification of the software architecture defines component instances and specifies interactions between those instances. For safety reasons engineers may need to duplicate or triplicate some components. This need is satisfied by the segregation of component models and component instances in the Mauve process.

The architecture specification consists in:

- creating and naming the components instances;
- linking each required method of the component instances to a provided method of another component instance (that satisfies signature consistency);
- connecting each input port of the component instances to an output port of another component (that satisfies data-type consistency); port connections are defined by connectors that specify the connection policy (buffered or not) and some parameters (size of the buffer, ...)

D. Deployment specification

In the proposed process, the architecture specification and the deployment specification are strictly separated for different reasons:

- experiments may lead to deploy the same software architecture on different hardware architectures;
- real-time parameters of some components may be adapted to their environment (e.g. effective sensor rate);
- schedulability analysis depends on hardware specification and codel implementation.

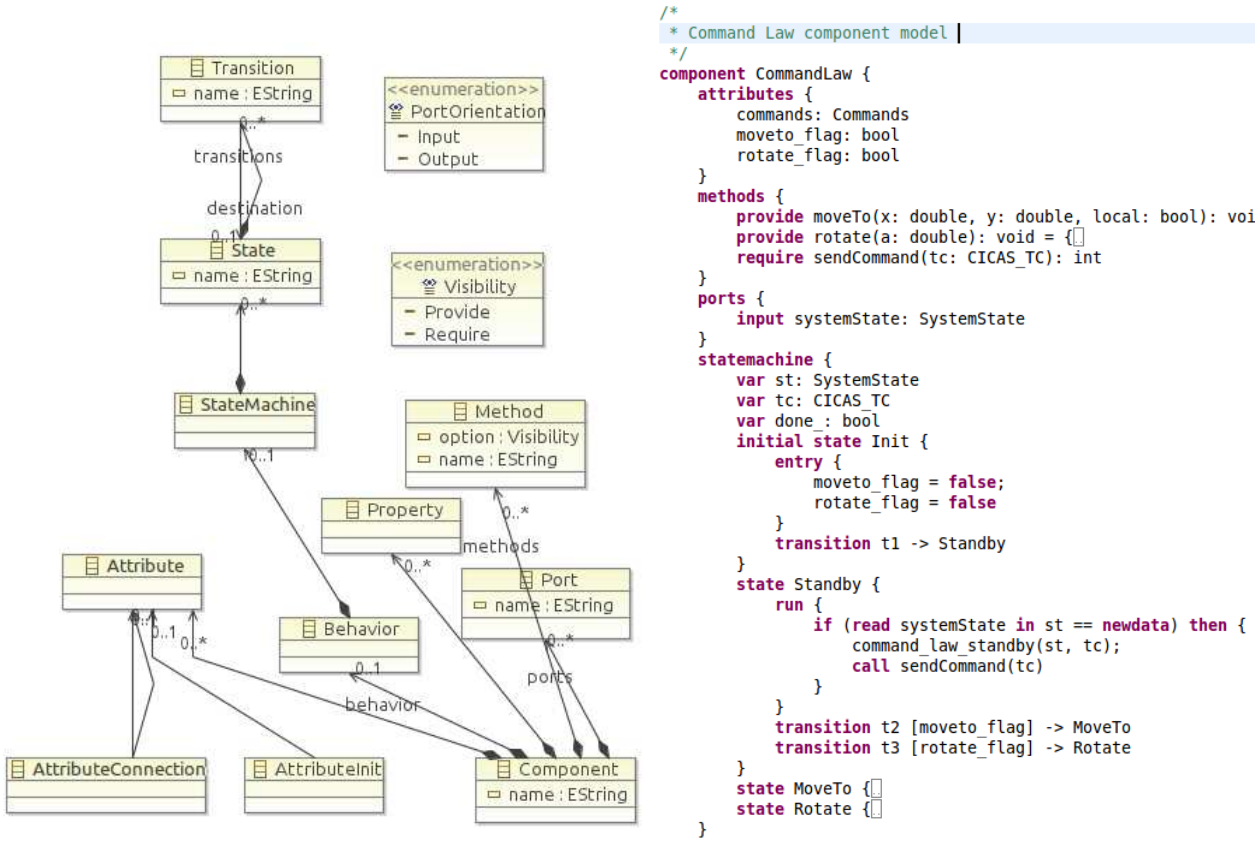
The deployment consists in specifying the worst case execution time (WCET) of each used codel and defining the real-time parameters of component instances (priority, period, deadline, ...). The WCET computation is accomplished using the Otawa framework [15] which statically analyses the assembled code mapped to an hardware architecture specification.

E. Real-time analysis

Each component instance is mapped into a real-time task. The tasks are executed on a real-time operating system [16] with a fixed priority scheduler. The schedulability analysis consists in computing the worst case response time (WCRT: worst delay between the task release and the end of its execution) of each task (i.e. component instance). A component instance meets its deadline if its WCRT is lower or equal to its deadline. We have defined a new method to compute with precision the WCRT of component instances. This method extends the well known schedulability analyses [17] by taking into account the state-machine of the real-time tasks.

F. Runtime deployment

For runtime deployment and execution, the Mauve specifications have not be mapped to an OS. Mauve



(a) The Mauve DSL

(b) Mauve concrete syntax

Fig. 2: Mauve component model

implements this mapping by generating code for the Orocos middleware [7]. This generation consists in C++ code for the component models, that is linked to the codels implementation, and in Orocos scripts for architecture and deployment specification.

IV. Conclusions

The design process based on the Mauve DSL provides best practices for robotic software development and deployment, by (1) clearly separating the *computation* (i.e. codels) from the architecture *configuration* (i.e. component models, architecture and deployment specifications), and (2) limiting the language to what can be analysed (e.g., some Orocos synchronization primitives[1] are not available in the Mauve language because they may lead to unsafe real-time behaviours or pessimistic analysis).

Finally, we plan to extend the runtime deployment process, by supporting other robotic middlewares, such as ROS [9], and investigating the generation of bare Xenomai tasks. These mappings will allow to further evaluate the real-time behaviours of the different runtime.

[1]namely Send/ClientThread for Orocos users

References

[1] K. Wiegers, "The Essential Software Requirement," in *Software Requirements*. Microsoft Press, 2003, ch. 1, pp. 3–25.

[2] D. Brugali and A. Shakhimardanov, "Component-Based Robotic Engineering (Part II)," *IEEE Robotics and Automation Magazine*, vol. 17, no. 1, pp. 100–112, 2010.

[3] A. Basu, M. Gallien, C. Lesire, T.-h. Nguyen, S. Bensalem, F. Ingrand, and J. Sifakis, "Incremental component-based construction and verification of a robotic system," in *Eureopean Conference on Artificial Intelligence (ECAI)*, Patras, Greece, 2008.

[4] A. Steck and C. Schlegel, "Towards quality of service and resource aware robotic systems through model-driven software development," in *International Workshop on Domain-Specific Languages and models for Robotic systems (DSLRob'10)*, 2010.

[5] C. Schlegel, A. Steck, D. Brugali, and A. Knoll, "Design Abstraction and Processes in Robotics: From Code-Driven to Model-Driven Engineering," in *International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR 2010)*, 2010, pp. 324–335.

[6] D. Alonso, C. Vicente-chicote, F. Ortiz, J. Pastor, and B. Alvarez, "V3CMM: a 3-View Component Meta-Model for Model-Driven Robotic Software Development," *Journal of Software Engineering for Robotics (JOSER)*, vol. 1, pp. 3–17, 2010.

[7] P. Soetens and H. Bruyninckx, "Realtime hybrid task-based control for robots and machine tools," in *International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain, 2005.

[8] A. Mallet, C. Pasteur, and M. Herrb, "GenoM3: Building middleware-independent robotic components," in *International Conference on Robotics and Automation (ICRA)*, Anchorage, AK, USA, May 2010, pp. 4627–4632.

[9] G. Bradski, K. Conley, B. Gerkey, E. Marder-Eppstein, M. Quigley, and M. Wise, "Tutorial on ROS," in *International Conference on Robotics and Automation (ICRA)*, Anchorage, AK, USA, 2010. [Online]. Available: http://www.ros.org/wiki/Events/ICRA2010Tutorial

[10] M. Radestock and S. Eisenbach, "Coordination in evolving systems," in *Trends in Distributed Systems (TreDS)*, 1996.

[11] BRIDE, "BRICS Integrated Development Environment," 2012. [Online]. Available: http://www.best-of-robotics.org/bride/

[12] R. Bischoff, T. Guhl, E. Prassler, W. Nowak, G. Kraetzschmar, H. Bruyninckx, P. Soetens, M. Haegele, A. Pott, P. Breedveld, J. Broenink, D. Brugali, and N. Tomatis, "BRICS - Best practice in robotics," in *International Symposium on Robotics (ISR)*, Munich, Germany, 2010.

[13] Proteus, "Platform for RObotic modeling and Transformations for End-Users and Scientific communities," 2012. [Online]. Available: http://www.anr-proteus.fr/

[14] A. Makarenko, A. Brooks, and T. Kaupp, "On the Benefits of Making Robotic Software Frameworks Thin," in *International Conference on Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

[15] C. Ballabriga, H. Cassé, C. Rochange, and P. Sainrat, "OTAWA: an Open Toolbox for Adaptive WCET Analysis," in *IFIP Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS)*, Waidhofen, Austria, 2010.

[16] P. Gerum, *Xenomai-Implementing a RTOS emulation framework on GNU/Linux*. GNU Free Documentation License, 2004.

[17] F. Zhang and A. Burns, "Schedulability analysis for real-time systems with EDF scheduling," *IEEE Transactions on Computers*, vol. 58, no. 9, 2009.